Some Fine Hierarchies on Relativized Time-Bounded Complexity Classes(LOGIC AND THE FOUNDATIONS OF MATHEMATICS)

AUTHOR(S):

TANAKA, Hisao; IZUMI, Masa-aki; TAKAHASHI, Nobuyuki

数理解析研究所講究録
516巻 1984年 53-78

Some Fine Hierarchies on Relativized Time-Bounded Complexity Classes

By

Hisao TANAKA, Masa-aki IZUMI and Nobuyuki TAKAHASHI
(田中尚夫, 和泉正明, 高橋信行)

§1. Introduction. One of the most important problems in the theory of computation is to find the precise relationship between the two kinds of computation : one is deterministic computation and the other is nondeterministic one. A fundamental open question is to determine whether or not

(1) $$P = NP,$$

where P is the class of languages accepted in polynomial time by deterministic Turing machines and NP is the counterpart of nondeterministic one. (Some authors use the terminology "accept" for nondeterministic machines and "recognize" for deterministic ones. Here we always use the word "accept".) Further, let P(k) [NP(k)] be the class of languages accepted in k-th order polynomial time by deterministic [resp. nondeterministic] Turing machines. We do not know whether or not

(2) $$P(k) = NP(k)$$

holds for any positive integer k either. Baker, Gill and

Solovay [1] (and others) show that the corresponding relativized question to (1) holds for some oracle sets on one hand and does not hold for some other oracle sets on the other hand. That is, they construct oracle sets A and B such that

(3) $\quad P(A) = NP(A) \quad \text{and} \quad P(B) \neq NP(B).$

This result gives a strong influence to methods which will be used to solve (1).

Now, first in this paper, we show that we can obtain analoguos results of (3) for question (2); namely, there can be various hierarchies on complexity subclasses of relativized NP (Theorems 1, 2 and 4). Secondly, we state that analogous results holds for another type of time complexity classes. Finally, we extend a result of Book, Wilson and Mei-Rui [3] to a class of higher level (Theorem 9).

(and results.)

§ 2. Preliminaries We mostly use standard terminology and notation for time-bounded complexity. See, e.g., [1]-[4]. Let $\Sigma = \{0,1\}$ be an alphabet. Σ^* denotes the set of all finite strings consisting of members of Σ. A subset L of Σ^* is called a language and we denote the complement of L by $\bar{L}$: $\bar{L} = \Sigma^* - L$. For $x \in \Sigma^*$, $|x|$ denotes the length of x. Our model for computation is the oracle Turing machines. An oracle Turing machine (abbreviated by OTM) is

We assume familiariy with them.

a multitape Turing machine with an query tape and with three special internal states called the query state $q_?$, the yes state q_Y and the no state q_N. When an OTM M is associated with an oracle set $X \subseteq \Sigma^*$, we denote it by M^X and call an OTM with oracle X. When an OTM M^X enters the state $q_?$, the machine asks the oracle X whether the string written on its query tape belongs to X. If the string is in X, then M^X enters q_Y, otherwise it enters q_N.

If the next-move-operation of M is single-valued, then we call M to be deterministic, otherwise M to be nondeterministic. Now, suppose that M^X runs on an input $x \in \Sigma^*$ and it halts after some running time. (Notice that we always consider OTM's that halt for every input.) If the final state in the computation is a special state called an accepting state we say M^X accepts x. Otherwise we say it rejects x. Let L be a language.L is accepted by an M^X (denoted by $L = T(M^X)$) if the following condition holds : For any string x $x \in L$ iff M^X accepts x.

Let ω be the set of all natural numbers, and let f be a function from ω to ω. An OTM M is f-time-bounded if every computation of M on any input x halts in fewer than $f(|x|)$ steps, whatever oracle X is used. Let $\mathcal{F}$ be a family of

such functions. An OTM M is said to be $\mathcal{F}$-time-bounded if M is f-time-bounded for some f in $\mathcal{F}$. We consider the following families of functions :

$$\mathcal{F}_P(k) = \{ p : p \text{ is a } k\text{-th order polynomial} \},$$

$$\mathcal{F}_P = \{ p : p \text{ is a polynomial} \} = \bigcup_{k=1}^{\infty} \mathcal{F}_P(k),$$

$$\mathcal{F}_{EXP}(k) = \{ \lambda n[2^{p(n)}] : p \text{ is in } \mathcal{F}_P(k) \},$$

$$\mathcal{F}_{EXP} = \{ \lambda n[2^{p(n)}] : p \text{ is in } \mathcal{F}_P \} = \bigcup_{k=1}^{\infty} \mathcal{F}_{EXP}(k),$$

where k is a positive integer. Let X be an oracle set. DTIME(X, f) [NTIME(X, f)] is the class of languages accepted by deterministic [resp. nondeterministic] f-time-bounded OTM's with oracle X. We consider the following classes of languages :

$$P(X, k) = \bigcup \{ DTIME(X, f) : f \text{ is in } \mathcal{F}_P(k) \},$$

$$P(X) = \bigcup_{k=1}^{\infty} P(X, k),$$

$$DEXT(X, k) = \bigcup \{ DTIME(X, f) : f \text{ is in } \mathcal{F}_{EXP}(k) \},$$

$$DEXT(X) = DEXT(X,1) = \bigcup \{ DTIME(X, f) : f \text{ is in } \mathcal{F}_{EXP}(1) \},$$

$$DEPT(X) = \bigcup_{k=1}^{\infty} DEXT(X, k),$$

and their corresponding nondeterministic classes NP(X,k), NP(X), NEXT(X, k), NEXT(X) and NEPT(X), respectively. Nonrelativized classes P, NP, DEXT and NEXT occur in literature, too.

We will use nonrelativized classes P(k) and NP(k), also.

For a positive real number α, $\lceil \alpha \rceil$ denotes the least integer larger than or equal to α. We often use $\exp(\alpha,\beta)$ instead of α^{β} when β has a rather complex expression. " $\lambda n O(f(n))$-time-bounded " means that $\lambda n[cf(n)]$-time-bounded for some constant $c > 0$. Similarly for the word " $O(f(n))$ steps". For simplicity, we often omit the symbol " λn " in the former case.

Now we shall state our theorems.

Theorem 1. There is an oracle set A such that for all $k > 0$ $P(A,k) \subsetneq NP(A,k) \subsetneq P(A,k+1)$. That is, we obtain the following hierarchy :

$$P(A,1) \subsetneq NP(A,1) \subsetneq P(A,2) \subsetneq NP(A,2) \subsetneq \ldots$$

(So, for this A, we have $P(A) = NP(A)$.)

Theorem 2. There is an oracle set B such that for all $k > 0$ $P(B,k) = NP(B,k)$, so that we obtain the following hierarchy :

$$P(B,1) = NP(B,1) \subsetneq P(B,2) = NP(B,2) \subsetneq \ldots.$$

(So, for this B, we have $P(B) = NP(B)$.)

Theorem 3. For any $k > 0$, if $P(k) = NP(k)$ then $P(k+1) = NP(k+1)$. So, e.g., if $P(1) = NP(1)$ then $P = NP$.

Of course, Theorem 3 can be relativized for every oracle set. In contrast with this theorem, we obtain the following theorem :

Theorem 4. For each $k > 0$, there is an oracle set C_k such that $P(C_k,k) \neq NP(C_k,k)$ but $P(C_k,k+1) = NP(C_k,k+1)$. So, we obtain the following hierarchy :

$(\forall i)[\ 1 \leq i \leq k \rightarrow P(C_k,i) \neq NP(C_k,i)\ ]\ \&\ (\forall j)[\ j > k \rightarrow P(C_k,j) = NP(C_k,j)\ ]$.

Theorems 1 to 4 can be extended to a case of exponential time-bounded complexity as follows :

Theorem 5. There is an oracle set D such that for all $k > 0$, $DEXT(D,k) \subsetneq NEXT(D,k) \subsetneq DEXT(D,k+1)$.

Theorem 6. There is an oracle set E such that for all $k > 0$ $DEXT(E,k) = NEXT(D,k)$.

Theorem 7. For any $k > 0$, if $DEXT(k) = NEXT(k)$ then $DEXT(k+1) = NEXT(k+1)$.

Theorem 8. For each $k > 0$, there is an oracle set F_k such that $DEXT(F_k,k) \neq NEXT(F_k,k)$ but $DEXT(F_k,k+1) = NEXT(F_k,k+1)$.

Next, Book [2] shows that for any oracle set X

(a) $P(X) = NP(X)$ implies $DEXT(X) = NEXT(X)$,

(b) $DEXT(X) = NEXT(X)$ implies $DEPT(X) = NEPT(X)$.

In contrast with (a), Book, Wilson and Mei-Rui [3] have shown that there is an oracle set W such that $P(W) \neq NP(W)$ but $DEXT(W) = NEXT(W)$. Here we show a counterpart of (b) for

their result. This is due to the first author.

Theorem 9. There is an oracle set G such that DEXT(G) $\neq$ NEXT(G) but DEPT(G) = NEPT(G).

§3. Proofs (1). In this section, we prove Theorems 1 and 2. First we show a lemma.

Lemma A. Let m and k be fixed positive integers. Let $f(n) = \exp(n, m+\frac{1}{k})$ and let $f'(n) = n^m \cdot \lceil \exp(n, \frac{1}{k}) \rceil$. Then there is a $c \cdot f$-time-bounded TM T for some constant c such that for any input w T outputs a string v (e.g., consisting of 0's only) whose length is $f'(|w|)$.

Sublemma. Let k be a fixed positive integer. Let $g(n) = \exp(n, 1+\frac{1}{k})$ and let $g'(n) = \lceil \exp(n, \frac{1}{k}) \rceil$. Then there is a $c \cdot g$-time-bounded TM T_1 for some constant c such that for any input w T_1 outputs a string v whose length is $g'(|w|)$.

Proof. Given an input w, let $d = g'(|w|)$. Then d is the least integer d such that $|w| \leq d^k$. T_1 successively generates strings of lengths $1^k, 2^k, 3^k, \ldots$ and finds a number d such that $d^{k-1} < |w| \leq d^k$. T_1 can do this in $O(d^{k+1})$ steps, since $1^k + 2^k + \ldots + d^k \leq d^{k+1}$. [1] So, in order to get a string v of length $g'(|w|)$, the number of entire steps performed by a TM T_1 with input w is bounded by $O(g(|w|))$, since

1) K. Kobayashi points out that $O(d^k)$ suffices for this upper bound.

$d^{k+1} < (\exp(|w|, \frac{1}{k})+1)^{k+1} < c\cdot\exp(|w|, 1+\frac{1}{k}) = c\cdot g(|w|)$ for some constant c. □

Proof of Lemma A. Let w be a given string. The TM T_1 in the sublemma with input w generates a string v_1 of length $g'(|w|)$ in fewer than $O(g(|w|))$ steps. Another TM T_2 with input w generates a string v_2 of length $|w|^m$ in fewer than $O(|w|^m)$ steps. We can easily define such a T_2. Then we can define a desired TM T by combining T_1 and T_2 so that T generates a string v of length $f'(|w|)$ in fewer than $O(f(|w|))$ steps, since

$$O(\exp(|w|, 1+\tfrac{1}{k})) + O(|w|^m) + O(\exp(|w|, \tfrac{1}{k})\cdot O(|w|^m) = O(\exp(|w|, m+\tfrac{1}{k})),$$

because of $m > 0$. □

Now let $\lambda ki[\langle k,i\rangle]$ be a recursive pairing function from $(\omega - \{0\}) \times \omega$ onto ω. If $e = \langle k,i\rangle$ we write $(e)_0 = k$ and $(e)_1 = i$. Let P_e [NP_e] be the e-th deterministic [resp. nondeterministic] polynomial-time-bounded OTM, and let f_e be its strict time-bound. We may assume, without loss of generality, that $f_e(n) = c_e\cdot n^k$, where $k = (e)_0$, $\exp(c_e, (k+\frac{1}{2})/k)$ is an even integer. For any oracle X and any $k > 0$, let

$$L_k(X) = \{ 0^m : \exists u \in X[|u| = m^k] \}.$$

Cleraly, $L_k(X)$ is in $NP(X,k)$.

Theorem 1. There is an oracle set A such that for all $k > 0$, $P(A,k) \neq NP(A,k) \neq P(A,k+1)$.

Proof. We construct an oracle set A in stages as in proofs described in [1],[3],[4]. Let A(s) be the set of all strings placed into A prior to stage s. Set $A(0) = A(1) = \emptyset$ (the empty set), and start at stage 1. At an odd stage we shall try to satisfy the condition $P(A,k) \neq NP(A,k)$ for some k, and at an even stage to satisfy the condition $NP(A,k) \subsetneq P(A,k+1)$ for some k. The index e of each machine P_e will be cancelled at some odd stage n_e (in order of the magnitude of e) when we ensure that P_e^A does not accept the language $L_k(A)$, where $k = (e)_0$.

Stage 2s+1. Let e be the first uncancelled index and put $e = \langle k,i \rangle$. Let $b = \max \{(a)_0 : a \leq e\}$. Note that $b \geq 1$. Suppose that the following conditions are satisfied :

(i) $\exp(2s+1, b/(k\cdot(b+\frac{1}{2})))$ is an (odd) integer. For simplicity, we denote it by m.

(ii) $\exp(2s+1, 1/(b+\frac{1}{2})) > f_{e-1}(\exp(n_{e-1}, b'/(j\cdot(b'+\frac{1}{2}))))$, where $j = (e-1)_0$ and $b' = \max \{(a)_0 : a \leq e-1\}$ if $e > 0$. If $e = 0$, then, as convention, the right-hand side of the inequality is 0.

(iii) $f_e(m) = c_e \cdot m^k < 2s+1 < 2^{m^k}$, where $m^k = \exp(2s+1, b/(b+\frac{1}{2}))$

Then, we execute the following procedure. Run the deterministic machine $P_e^{A(2s+1)}$ on the string 0^m. If the machine rejects 0^m, then we add to A a string u of odd length m^k not queried during the computation : $A(2s+2) = A(2s+1) \cup \{u\}$. By (iii), such a string u exists. If 0^m is accepted let $A(2s+2) = A(2s+1)$. Cancel the index e at this stage in either case, and set $n_e = 2s+1$.

If at least one of the conditions (i)-(iii) does not hold, then we do nothing and set $A(2s+2) = A(2s+1)$. Clearly every index eventually be cancelled.

Stage 2s+2. Let e be the <u>last</u> index that was cancelled at an odd stage before 2s+2. If there is no such e, we skip this stage and set $A(2s+3) = A(2s+2)$. If $e \geq 0$ let $d = \max\{(a)_0 : a \leq e\}$. Consider a string y such that

(1) $y = 0^k 1 0^i 1 x 1 0^n$ & $k \leq d$ & $2s+2 = |y|$

$$= \exp(c_{\langle k,i\rangle}, \frac{k+1/2}{k}) \cdot |x|^k \cdot \lceil |x|^{\frac{1}{2}} \rceil .$$

for some k, i, n $\in \omega$ and $x \in \Sigma^*$.

For such an y, run $NP_{\langle k,i\rangle}^{A(2s+2)}$ on the string x. Note that length of any string queried during any computation of this machine on x is less than

$$c_{\langle k,i\rangle}|x|^k \leq \exp(2s+2, \frac{k}{k+1/2}) \leq \exp(2s+2, \frac{d}{d+1/2}) < 2s+2.$$

If any computation of $NP^{A(2s+2)}_{\langle k,i\rangle}$ accepts x, then we place y into A. Otherwise, we do not so. Let A'(2s+3) be the set of all strings placed into A by performing the above procedure for all y satisfying (1), and set A(2s+3) = A(2s+2) $\cup$ A'(2s+3). If there is no y satisfying (1), then we do nothing and let A(2s+3) = A(2s+2). So, information about nondeterministic computation of $NP^{A}_{\langle k,i\rangle}$ is encoded into strings of A of the form y (of even length) in (1).

Define A by $A = \bigcup_{s=0}^{\infty} A(s)$. We show that A is a desired oracle set.

Claim 1. The string u placed into A at stage 2s+1 is not queried in any computation performed at any ealier stage. [Proof. Length of any string queried at stage n_{e-1} is less than |u|, since by (ii)

$$f_{e-1}(\exp(n_{e-1}, \frac{b'}{j\cdot(b'+1/2)})) < \exp(2s+1, \frac{1}{b+1/2}) < m^k.$$

By similar computation, we see that |u| is larger than length of any string queried at any earlier odd stage. Next, at a stage $2t < n_{e-1}$, length of any string queried in any computation is less than |u|, since

$$\exp(2t, \frac{d'}{d'+1/2}) < \exp(n_{e-1}, \frac{b'}{b'+1/2}) < \exp(2s+1, \frac{1}{b+1/2}) < |u|,$$

where $d' = \max\{(a)_0 : a \leq e''\}$ ($\leq b'$) and e" is the last (i.e., largest) index cancelled at an odd stage before 2t. (If

$e = 0$ we do not have to consider this situation.) Length of any string queried at any stage $2t$ between n_{e-1} and $2s+1$ is less than $|u|$ also,since

$$\exp(2t, \frac{d}{d+1/2}) = \exp(2t, \frac{b'}{b'+1/2}) < \exp(2s+1, \frac{b}{b+1/2}) = |u| ,$$

where $d = \max\{(a)_0 : a \leq e-1\} = b'$. Thus, u is not queried at any earlier stage $< 2s+1$.]

Claim 2. When an even stage $2s+2$ is executed, any such string y is not queried at any ealier stage. [Proof. Length of any string queried at any ealier odd stage $2s'+1 < 2s+2$ is less than $|y|$, since $f_{e'}(m') < 2s'+1 < |y|$, where e' is the cancelled index at stage $2s'+1$ and m' is the m defined in (i) with $2s'+1$ instead of $2s+1$. Length of any string queried at any eariler even stage $2s' < 2s+2$ is less than $|y|$, since it is less than $2s'$. So, y is not queried at any earlier stage.]

Claim 3. For all $k \geq 1$, $NP(A,k) \subsetneq P(A,k+1)$. [Proof. Let $k \geq 1$ be given. Let L be an arbitrary language in $NP(A,k)$. Then, there is an i such that $L = T(NP^A_{\langle k,i\rangle})$. We construct a deterministic $O(n^{k+1/2})$-time-bounded OTM M with oracle A as follows : Let $e' = \langle k,i\rangle$. Since e' is eventually cancelled, $n_{e'} = 2s'+1$ is determined. Given an input x, first M generates the string y such that

(2) $y = 0^k 1 0^i 1 x 1 0^n$ and $2s'+1 < |y| = \exp(c_{\langle k,i\rangle}, \frac{k+1/2}{k}) \cdot |x|^{k\lceil |x|^{\frac{1}{2}} \rceil}$.

By Lemma A, this is done in $O(|x|^{k+1/2})$ steps. There are only finitely many (possibly zero) x's for which there is no y such that (2) holds. So, we can make a finite table so that M accepts x if and only if x is in L for such x's. Now, let $2s+2 = |y|$. Since $2s+2 > n_{e'}$, letting e be the last index cancelled before stage 2s+2, we have $e' \leq e$. So, $k \leq d$, where $d = \max \{ (a)_0 : a \leq e \}$. Hence, (1) holds for these 2s+2, y, k, i, n, and x. If y is in A, then M accepts x, otherwise M rejects x. Thus, the constructed machine M with oracle A accepts x iff y is in A iff $NP^{A(2s+2)}_{\langle k,i \rangle}$ accepts x iff $NP^{A}_{\langle k,i \rangle}$ accepts x [by Claim 2] iff x is in L. That is, M^A accepts L. Obviously M is a deterministic $O(n^{k+1/2})$-time-bounded OTM. So, L is in P(A, k+1/2), where P(A, k+1/2) is the class of languages accepted by deterministic $O(n^{k+1/2})$-time-bounded OTM's with oracle A. By the Hierarchy Theorem, $P(A,k+1/2) \subsetneq P(A,k+1)$. Therefore $NP(A,k) \subsetneq P(A,k+1)$.]

Claim 4. $L_k(A)$ is not in P(A,k) and hence $P(A,k) \neq NP(A,k)$. [Proof. Let i be arbitrary ≥ 0, and consider $e = \langle k,i \rangle$. Then n_e is determined, and let $m = \exp(n_e, \frac{b}{k(b+1/2)})$, where $b = \max \{ (a)_0 : a \leq e \}$. Then P^A_e rejects 0^m iff $P^{A(n_e)}_e$ rejects 0^m [by Claim 1] iff $\exists u \in A[|u| = m^k]$ iff $0^m \in L_k(A)$. So, P^A_e does not accept $L_k(A)$. Hence $L_k(A)$ is

not in P(A,k).]

Claims 3 and 4 prove Theorem 1. □

Theorem 2. There is an oracle set B such that for all $k \geq 1$ $P(B,k) = NP(B,k)$.

Proof. Let B(s) be the set of all strings placed into B before stage s, and let $B(0) = \emptyset$.

Stage s. Consider a string y such that

(3) $y = 0^k 1 0^i 1 x 1 0^n$ and $s = |y| = c_{\langle k,i \rangle} |x|^k$ for some $k \geq 1$, $i, n \in \omega$ and $x \in \Sigma^*$.

Run $NP^{B(s)}_{\langle k,i \rangle}$ on x. Length of strings queried in any computation of $NP^{B(s)}_{\langle k,i \rangle}$ on x is less than s. If x is accepted we place y into B. Otherwise we do nothing. We execute this procedure for every y satisfying (3), and let B'(s+1) be the set of all strings added to B at this stage s. Let $B(s+1) = B(s) \cup B'(s+1)$. If there is no such y we let $B(s+1) = B(s)$.

Define $B = \bigcup_{s=0}^{\infty} B(s)$. Then B is a desired set. □

§4. Proofs (2). In this section we prove Theorems 3 and 4.

Theorem 3. For any $k \geq 1$, if $P(k) = NP(k)$, then $P(k+1) = NP(k+1)$.

Proof. Basic idea is due to Book [2]. Let $k \geq 1$ be fixed, and assume $P(k) = NP(k)$. For $r = k$ or $k+1$, let $P_{\langle r,i \rangle}$ [resp. $NP_{\langle r,i \rangle}$] be the i-th deterministic [resp. non-

deterministic] r-th-order-polynomial-time-bounded TM with its strict time-bound $f_{\langle r,i\rangle}$, where $f_{\langle r,i\rangle}(n) = c_{\langle r,i\rangle}\cdot n^r$. This time we assume that $\exp(c_{\langle k+1,i\rangle}, 1/k)$ is a positive integer for every i. Now let L_1 be an arbitrary language in NP(k+1). Take an i so that $L = T(NP_{\langle k+1,i\rangle})$. Define L_2, as in [2; p.225], by

$$L_2 = \{\, 0^m1w : w \in L_1 \;\&\; |0^m1w| \geq \exp(c_{\langle k+1,i\rangle}, 1/k)\cdot|w|^{\lceil |w|^{1/k}\rceil} \,\}.$$

Using $NP_{\langle k+1,i\rangle}$, we can define a nondeterministic $O(n^k)$-time-bounded TM M_2 which accepts L_2. Hence L_2 is in NP(k). Since P(k) = NP(k), there is an index j such that $L_2 = T(P_{\langle k,j\rangle})$. Using $P_{\langle k,j\rangle}$ we can define a deterministic $O(n^{k+1})$-time-bounded TM M which accepts L_1. [For, on a given input w M first produces $w10^m$ such that $|0^m1w| = \exp(c_{\langle k+1,i\rangle}, 1/k)\cdot|w|^{\lceil |w|^{1/k}\rceil}$. By Lemma A , such action needs at most $\text{constant}\cdot|w|^{1+1/k}$ steps only. Then M simulates $P_{\langle k,j\rangle}$ on 0^m1w . If M_2 accepts 0^m1w then M accepts w, otherwise M rejects w. This simulation needs (at most) $\text{constant}\cdot c_{\langle k+1,i\rangle}|w|^{k+1}$ steps only.] Therefore NP(k+1) is contained in P(k+1). □

Theorem 4. For each $k \geq 1$, there is an oracle set C_k such that $P(C_k,k) \neq NP(C_k,k)$ but $P(C_k,k+1) = NP(C_k,k+1)$.

Proof. Let $k \geq 1$ be fixed and we use subenumerations $\{P_e : (e)_0 = k \text{ or } k+1\}$ and $\{NP_e : (e)_0 = k \text{ or } k+1\}$, where

P_e and NP_e are OTM's stated in the preceding section. This time, we assume that $\exp(c_{\langle k+1,i\rangle}, 1/(k+1))$ is a positive integer for every i. As before, C is constructed at stages. Let C(s) be the set of all strings placed into C before stage s. At each stage, some strings (possibly nothing) will be put in C and some other strings (possibly nothing) will be reserved for the complement $\overline{C}$. At each stage, if a string is neither placed into C nor reserved for $\overline{C}$ yet, we say it to be unreserved at this stage. As before, the index i of each OTM $P_{\langle k,i\rangle}$ will be cancelled at some odd stage when we ensure that $P_{\langle k,i\rangle}$ does not accept the language $L_k(C)$, which is defined in the preceding section with X = C. Let C(0) = $\emptyset$, $n_{-1} = 0$ and $f_{\langle k,-1\rangle}(0) = 0$.

Stage 2s. Consider the following condition

(1) $$s^{k+1} \cdot 2^s < 2^{2(s-2)}$$

If (1) does not hold, we skip this stage and let C(2s+1) = C(2s). Suppose (1) holds. (Clearly all sufficiently large s's satisfy (1).) Consider strings y such that

(2) $y = 0^i 1x10^n$ & $|y| = s$ for some i, $n \in \omega$ and $x \in \Sigma^*$.

Let

(3) $y_1, y_2, \ldots, y_r$

be an enumeration of all such y's. We put $y_j = 0^{i(j)}1x_j10^{n(j)}$ for $1 \leq j \leq r$. Set $C_0(2s) = C(2s)$ and consider y_j. For

simplicity, we write $y_j = 0^i 1x1 0^n = y$. Run $NP^{C_{j-1}(2s)}_{\langle k+1,i\rangle}$ on x in fewer than n^{k+1} steps. Note that the length of a string queried in any computation of this machine on x is less than s^{k+1}. If x is accepted within $n^{k+1}-1$ steps, we add to $C_j(2s)$ the even string $y0^m$, where $m = |y|^{k+1}$. Further we reserve all unreserved strings queried in this accepting computation for $\overline{C}$. If no computation of this machine accepts x in fewer than n^{k+1} steps, then determine whether the addition of some unreserved strings to $C_{j-1}(2s)$ will lead to acceptance. (This and the following ideas are due to [3; p. 579].) If so (case (a)), then place those unreserved strings queried in an accepting computation into C and $\overline{C}$ appropriately so that acceptance is preserved. Place $y0^m$ into C. Let $C_j(2s)$ be the set obtained from $C_{j-1}(2s)$ by adding all such strings as above. If not (case (b)), then we only do reservation of $y0^m$ for $\overline{C}$. Let $C(2s+1) = C_r(2s)$ and go to the next stage. Note that any computation of $NP^{C_j(2s)}_{\langle k+1,i\rangle}$ on x is not affected by any possible later addition to C. And for the case of the above illustration

(4) $y0^m$ is in C iff $NP^{C_j(2s)}_{\langle k+1,i\rangle}$ accepts x in fewer than n^{k+1} steps.

If there is no y satisfying (2), let $C(2s+1) = C(2s)$.

Stage 2s+1. Let i be the first uncancelled index. Sup-

pose the following 4 conditions hold :

(i) $f_{\langle k,i-1\rangle}(n_{i-1}) < s + s^{k+1}$ ($= |y0^m|$ for $|y| = s$, where $m = |y|^{k+1}$),

(ii) $f_{\langle k,i-1\rangle}(n_{i-1}) < (2s+1)^k$,

(iii) $f_{\langle k,i\rangle}(2s+1) = c_{\langle k,i\rangle}(2s+1)^k < (s+1) + (s+1)^{k+1} < (2s+1)^{k+1}$,

(iv) $(2s+1)^{k+1} + 2^{2s} < 2^{(2s+1)^k}$.

Then we cancel i and set $n_i = 2s+1$. Run $P^{C(2s+1)}_{\langle k,i\rangle}$ on $z = 0^{2s+1}$. We reserve all unreserved strings queried in the computation for $\overline{C}$. If the machine rejects z, then we take a string u of length $(2s+1)^k$ such that u is not queried in the computation and is unreserved at the beginning of this stage. (Existence of such a string u is proven in Claim 2.) Set $C(2s+2) = C(2s+1) \cup \{u\}$. If the machine accepts z, then we reserve all strings of length $(2s+1)^k$ for $\overline{C}$ and let $C(2s+2) = C(2s+1)$.

If one of the conditions (i) - (iv) does not hold, then we skip this stage and let $C(2s+2) = C(2s+1)$. Clearly each index i will eventually be cancelled.

Define C by $C = \bigcup_{s=0}^{\infty} C(s)$. We show that C is a desired oracle set.

Claim 1. Any string $w = y0^m$, where $m = |y|^{k+1}$, placed

into C or reserved for $\overline{C}$ at an even stage 2s was not queried at any earlier stage. Hence, w is unreserved at the beginning of stage 2s. Proof is done by using (i) and (iii).
[Length of any strings queried at any earlier even stage 2s' ($< 2s$) is less than $(s')^{k+1} \leq |w|$. Let 2s"+1 be the latest odd stage which was executed before 2s, and let i be the cancelled index at this stage. By (i) and (iii) $f_{\langle k,i-1\rangle}(n_{i-1}) < |w|$ and $f_{\langle k,i\rangle}(2s"+1) < |w|$. So w is longer than any string queried at any earlier stage.]

Claim 2. When an odd stage 2s+1 is executed, there is such a string u. Note that even though u had been queried in some computation for the case (b) at an earlier even stage 2s' ($< 2s+1$) the case (b) remains true after adding u to C.
[Proof. By (ii), we do not have to consider any earlier odd stage. So, we evaluate the number of strings reserved at earlier even stages. Let $2s' < 2s+1$. At stage 2s', for each string $y = 0^i 1 x 1 0^n$ such that $|y| = s'$ the number of strings reserved at this stage is less than $n^{k+1} < (s')^{k+1}$. So the number of strings reserved with respect to all such y's is less than $(s')^{k+1}$.(the number of such x's) $< (s')^{k+1} \cdot 2^{s'} < 2^{2(s'-2)}$ by (1). Hence the entire number of strings reserved at all earlier even stages $\leq 2s$ is less than $\sum_{s'=s_0}^{s} 2^{2(s'-2)} < 2^{2s}$, where s_0 is the least s_0 such that $s_0^{k+1} \cdot 2^{s_0} < 2^{2(s_0-2)}$

holds. The number of strings queried at stage 2s+1 is less than $(2s+1)^{k+1}$ (by (iii)). So by (iv), their sum is less than $2^{(2s+1)^k}$. Every string placed into C at an even stage is of even length.]

By using Claim 2 we can show $P(C,k) \neq NP(C,k)$ as before. Finally we show $P(C,k+1) = NP(C,k+1)$. Let L be an arbitrary language in $NP(C,k+1)$ and let i be such that $L = T(NP^C_{\langle k+1,i\rangle})$. We define a deterministic $\lambda n O(n^{k+1})$-time-bounded OTM M with oracle C as follows : Given an input x, M produces a string y such that $y = 0^i 1 x 1 0^n$, where $n = \exp(c_{\langle k+1,i\rangle}, 1/(k+1)) \cdot |x|$. This is done in fewer than $O(|x|)$ steps. Put $s = |y|$, and consider the condition (1). We execute stage 2s. Let the above y be the j-th member in the enumeration (3) : $y = y_j$. M produces a string $y0^m$, where $m = |y|^{k+1}$. This is done in fewer than $O(|x|^{k+1})$ steps. Then M asks the oracle C if $y0^m$ is in C. If the answer is yes, then M accepts x. Otherwise M rejects x. By (4) and Claim 1, x is in L iff $NP^C_{\langle k+1,i\rangle}$ accepts x iff $NP^{C_j(2s)}_{\langle k+1,i\rangle}$ accepts x iff $NP^{C_j(2s)}_{\langle k+1,i\rangle}$ accepts x in fewer than n^{k+1} (= $c_{\langle k+1,i\rangle} \cdot |x|^{k+1}$)steps iff $y0^m$ is in C. Therefore M accepts L. (Of course , if neccesary, we use a finite table, too.) Clearly M is a deterministic $\lambda n O(n^{k+1})$-time-bounded OTM. So, L is in P(C,k+1) and hence $NP(C,k+1) \subseteq P(C,k+1)$. □

Proofs of Theorems 5, 6, 7 and 8 are entirely similar to those of Theorems 1, 2, 3 and 4. So we omit them.

§5. Proofs (3). In this section, we show Theorem 9. Let DE_i [resp. NE_i] be the i-th deterministic [resp. nondeterministic] $\mathcal{F}_{EXP}(1)$-time-bounded OTM with its strict time-bound g_i, where $g_i(n) = 2^{d_i \cdot n}$, $d_i > 0$. Further, let DEP_i [resp. NEP_i] be the i-th $\mathcal{F}_{EXP}$-time-bounded OTM with its strict time-bound $\lambda n[\, 2^{p_i(n)} \,]$, where p_i is a polynomial. Now, first we state a proof of (b) stated in the last paragrph of §1.

Proposition. (Book [2]) For any oracle set X, if $DEXT(X) = NEXT(X)$ then $DEPT(X) = NEPT(X)$.

Proof. Let, for any oracle X,

$$K(X) = \{\, 0^i 1 x 1 0^n : \text{some computation of } NEP_i^X \text{ accepts } x \text{ in fewer than } 2^n \text{ steps} \,\}.$$

Clearly, K(X) is in NEXT(X) and hence $NEPT(X) \subseteq P(K(X))$ as in [1; Proof of Lemma 1 on p.433]. Suppose $DEXT(X) = NEXT(X)$. Then K(X) is in DEXT(X) and so there is an index i such that DE_i^X accepts K(X). To show $NEPT(X) \subseteq DEPT(X)$, let L be an arbitrary language in NEPT(X). Then there is an index j such that $P_j^{K(X)}$ accepts L. Define a deterministic $\mathcal{F}_{EXP}$-time-bounded OTM M with oracle X which accepts L : Given an input

x, M first simulates the computation of $P_j^{K(X)}$ on x. If a string w is queried in the computation, then using oracle X M simulates the computation of DE_i^X on w and decides whether w is in K(X). The latter simulation can be done in fewer than $2^{d_i \cdot |w|}$ steps. M accepts x if $P_j^{K(X)}$ accepts x. So, M accepts L. Length of such a string w is less than $p_j(|x|)$ and the number of queried strings is less than $p_j(|x|)$. So, the entire steps of the computation of M on x is bounded by $2^{p(|x|)}$ for some polynomial p. Hence M is a deterministic $\mathcal{F}_{EXP}$-time-bounded OTM. Therefore L is in DEPT(X). □

In contrast with this proposition we have

Theorem 9. There is an oracle set G such that DEXT(G) ≠ NEXT(G) but DEPT(G) = NEPT(G).

Proof. Let G(s) be the set of all strings placed into G before stage s and let G(0) = ∅. At some stages we reserve some strings for $\overline{G}$ and the index e of each DE_e is cancelled at some stage n_e when we ensure that DE_e^G does not accept the language

$$L_{EX}(G) = \{ 0^n : \exists u \in G[\, |u| = 2^n \,] \}.$$

Clearly $L_{EX}(G)$ is in NEXT(G). Let $n_{-1} = 1$ and $g_{-1}(0) = 1$.

Stage 2s. Consider strings y such that

(1) $y = 0^i 1 x 1 0^n$ and $2s = |y| = 2^{p_i(|x|)}$ for some i, n ∈ ω and x ∈ Σ*.

Run $NEP_i^{G(2s)}$ on x. If it accepts x, then we take a string yw such that

(2) $|w| = p_i(|x|)^2 + \varepsilon$, where $\varepsilon = 0$ or 1,

(3) $|yw|$ is odd and

(4) yw is not reserved for $\overline{G}$, yet.

And we add yw to G. Such a string yw exists. (See Claim 2.) If $NEP_i^{G(2s)}$ rejects x we do nothing. Let G'(2s+1) be the set of all strings placed into G at this stage after performing the above procedure for every y for which (1) holds, and let $G(2s+1) = G(2s) \cup G'(2s+1)$. If there is no such y let $G(2s+1) = G(2s)$.

Stage 2s+1. Let e be the first uncancelled index at the beginning of this stage. Suppose the following 4 conditions are satisfied :

(i) $2s+1 > g_{e-1}(\log n_{e-1})$,

(ii) $\log(2s+2)$ is an even number,

(iii) There is no string of length larger than 2s+1 which is reserved for $\overline{G}$ before this stage,

(iv) $g_e(\log(2s+2)) < 2^{(\log(2s+2))^2} < 2^{2s+2}$.

Then we cancel e at this stage and put $n_e = 2s+1$. Run $DE_e^{G(2s+1)}$ on the string $z_{2s+1} = 0^{\log(2s+2)}$. We reserve for $\overline{G}$ all strings of lengths larger than 2s+1 negatively queried in the computation. If $DE_e^{G(2s+1)}$ rejects z_{2s+1}, then we

choose a string u of length 2s+2 not queried in the computation and add it to G(2s+1) to make G(2s+2). By (iv), such a string u exists. If $DE_e^{G(2s+1)}$ accepts z_{2s+1} let G(2s+2) = G(2s+1). If at least one of (i) - (iv) is not satisfied, then we do nothing and put G(2s+2) = G(2s+1). Clearly each index e will eventually be cancelled.

Let $G = \bigcup_{s=0}^{\infty} G(s)$. G is a desired oracle set :

Claim 1. When an odd stage 2s+1 is executed, the following condition holds, too :

(v) For any ε, y, i, x and n, if $\varepsilon = 0$ or 1 and $y = 0^i 1 x 1 0^n$ and $2s+1 \leq |y| \leq 2^{p_i(|x|)}$, then there is a string w such that $|w| = p_i(|x|)^2 + \varepsilon$ and |yw| is odd and $DE_e^{G(2s+1)}$ does not query yw in the computation on z_{2s+1}.

[Proof is by (iv).]

Claim 2. When an even stage 2s is executed, such a yw exists and this string is not queried in any computation performed at any earlier stage. [Proof. Let $y = 0^i 1 x 1 0^n$ with $2s = |y| = 2^{p_i(|x|)}$ be any string taken at stage 2s. Let 2s'+1 be the last odd stage executed before 2s, and let e be the cancelled index at 2s'+1. By Claim 1, there is a string w such that $|w| = p_i(|x|)^2 + \varepsilon$, |yw| is odd and $DE_e^{G(2s'+1)}$ does not query yw in the computation on $z_{2s'+1}$.

since only strings of lengths less than 2s'+1 are queried at odd stages $< 2s'+1$ (because of (i) with s'), yw is not reserved for $\overline{G}$ yet. Moreover, only strings of lengths less than 2s are queried at even stages $< 2s$. So yw is not queried at any earlier stage.]

Claim 3. When an odd stage 2s+1 is executed, such a string u is not queried at any earlier stage. [Proof is by (iii).]

As before, we see $L_{EX}(G)$ is not in DEXT(G), by using Claim 3. Finally, we show NEPT(G) is contained in DEPT(G). Let L be in NEPT(G) and let i be such that $L = T(NEP_i^G)$. We define a deterministic $\mathcal{F}_{EXP}$-time-bounded OTM M with oracle G which accepts L. Given x, M first produces a string y such that

(5) $y = 0^i 1 x 1 0^n$ & $|y| = 2^{p_i(|x|)}$

Let $|y| = 2s$. Then M produces a string w such that $|w| = p_i(|x|)^2 + \varepsilon$ ($\varepsilon = 0$ or 1) and such that $|yw|$ is odd. M accepts x iff yw is in G. So, by Claim 2, M accepts x iff $NEP_i^{G(2s)}$ accepts x iff NEP_i^G accepts x iff x is in L. Hence L is accepted by M^G. Guessing such a string w can deterministically be done in fewer than const $2^{p_i(|x|)^2+1}$ steps. So, using oracle G M can deterministically decide whether it accepts x in fewer than $2^{p(|x|)}$ steps for some

polynomial p. Hence L is in DEPT(G) . Consequently NEPT(G) is contained in DEPT(G).

A language on a one-letter alphabet is called a tally language. It is known,by Book, that if DEXT = NEXT then P can not be separated from NP by any tally language. Contrast with this, we have :

Collorary. There is an oracle set G such that DEPT(G) = NEPT(G) but DEXT(G) is separated from NEXT(G) by a tally language.

References

[1] T.Baker, J.Gill and R.Solovay, Relativizations of P =? NP question, SIMA J. Comput., 4 (1975), 431-442.

[2] R.Book, Compairing complexity classes, J. Comput. Syst. Sci., 9 (1974), 213-229.

[3] R.Book, C.Wilson and Xu Mei-Rui, Relativizing time, space, and time-space, SIAM J. Comput., 11 (1982), 571-581.

[4] C.Kintala and P.Fischer, Refining nondeterminism in relativized polynomial-time bounded computation, Ibid. 9 (1980), 46-53.

The first and second author : College of Engineering
Hosei University

The third author : Yokohama College of Commerce